# Sequence Labeling: Chinese Word Segmentation and Named Entity Recognition

李大為 **Da-Wei Lee**
Peking University
1701210963@pku.edu.cn

## 1 Introduction

The goal of this subject is to learn the sequence labeling in the natural language processing field.

There are two classic and basic task in natural language processing, especially in Chinese, the word segmentation and named entity recognition.

In Chinese, we need word segmentation to seperate words, even the little difference of splitting point may possibly influence whole meaning of entire sentence.

And extract named entities from an article or web page can helps us quickly summary the subject. Sometimes we can also find the relationship between named entities that helps us to form the connection between them.

### 1.1 Word Segmentation

In word segmentation[1] task, we can reduce it to a 4-class classification problem.

The four classes:

- B: Begin of a token
- M: Middle of a token
- E: End of a token
- S: single character as a token

Thus a seperated (labeled) dataset can be transform to the trainable descripiton - a word with a label, a sentence as a sequence.

For example: 小明 吃 冰淇淋 ⇒ (小, B), (明, E), (吃, S), (冰, B), (淇, M), (淋, E).

**Evaluation**

The evaluation of the word segmentation task is not that simple, because the "word-level" measurement means nothing. So we need "entity-level" metrics.

There is an approach called the Golden Standard, the classic evaluation method is given by SIGHAN Bakeoff 2005[2] .

We mark the words' start position and the end position then compare to the golden standard position. Then we can get the following counts:

- $N$: Golden standard's word count
- $e$: Error (not in golden standard) word count
- $c$: Correct (in golden standard) word count

And use these numbers we can calculate precision ($P = \frac{c}{c+e}$), recall ($R = \frac{c}{N}$) and f1 score ($F_1 = \frac{2\times P\times R}{P+R}$). We can also get "error rate" ($ER = \frac{e}{N}$) as one of the metrics.

**Given Data**

Here is the summary of the given training data.

- Sentences in training data: 95304

[1] https://en.wikipedia.org/wiki/Text_segmentation

[2] http://sighan.cs.uchicago.edu/bakeoff2005/

- Total unique word: 4743
- Max sentence (sequence) length: 165
- Classes to classify: 4

## 1.2 Named Entity Recognition

Named entity recognition[3] task is very similar with word segmentation. We can also reduce it into a sequence labeling or a classification problem.

In our data we have three tags PER, LOC, ORG represent person, location, organization respectively. Each of them like word segmentation has begin and middle (but without end). Here is the labels:

- B-PER: Begin of a person entity
- I-PER: In a person entity
- B-LOC: Begin of a location entity
- I-LOC: In a location entity
- B-ORG: Begin of a organization entity
- I-ORG: In a organization entity
- N: Not an entity (sometimes use O)

The given dataset is already transfered to the trainable format - a word with a label and sentences are sepreated with double new line character.

So in this task we can save the effort of transfer between trainable format and the final output.

### Evaluation

The evaluation of named entity recognition is also a little tricky. And we need to also evaluate on entity-level.

Because, in the dataset, the data is almost labeled with N tag. If use word-level metrics we will get about 90% precision even we predict N all the times.

There are plenty of evaluation metrics, such as CoNLL-2003[4] use only precision, recall and f1 score; SemEval-2013 Task 9-1[5] use four advanced metrics (strict, exact, partial, type) to calculate precision, recall, f1 score.

And I'll use SemEval's metrics later on.

### Given Data

Here is the summary of the given training data.

- Sentences in training data: 36334
- Total unique word: 4378
- Max sentence (sequence) length: 374
- Classes to classify: 7

# 2 Approach

In this paragraph I'll introduce the approaches including tricks and models that I attempt to solve this task.

## Data Preprocessing

Because the sentence length of the training data are differ. But the model's input should be a fixed shape. Thus I have padded all the sentences into the maximum sentence length among both training data and test data. In the meanwhile, it is necessary to record the actual sequence length, when we will need to mask the rest of the part out when evaulation during training phase.

The padding is using an padding character, it is shared with Out-of-vocabulary character. The OOV happened when testing the test data that is possible appear words not in training set. Thus I have to replace it with OOV and refill it back when generating output file.

I tried to encode the words with one-hot encoding. And in the first place found that it is not possible to transfer entire dataset at once. By calculating the data size it will need more than 500GB memory which is

[3] https://en.wikipedia.org/wiki/Named-entity_recognition

[4] https://www.aclweb.org/anthology/W03-0419

[5] https://www.cs.york.ac.uk/semeval-2013/accepted/76_Paper.pdf

not practical. Thus, I encode sentences only when needed (i.e. per batch).

### 2.1 CRF

CRF [2], or Conditional Random Field[6] , is a undirected probabilistic graphical model, usually be used in sequence labeling task.

Choose CRF is better than HMM, Hidden Markov Model[7] , in the current case. That is because by given all the data and labels, discriminative model will perform better than generative model. Due to their feature function, the CRF consider the global but the (first-order) HMM impose a dependency only to the previous element. Thus in general CRF is more powerful than HMM.

Training the CRF that is learning the conditional distributions between the true sequence labels and features.

In practice, I use Viterbi algorithm to decode the transition matrix. And train the model with the log likelihood.

### 2.2 BiLSTM with CRF

I found that the feature selection is very important to the CRF. One-hot encoding is not good at representing a word's meaning. The BiLSTM with CRF [1] approach is kind of a solution.

When using bidirectional RNN, we can consider it as a dynamic embedding layer that pack the meaning of a word. Use this as the input of the CRF will get much better result.

And because bidirectional, it can better extract the context among a word, compare with simple RNN or other naive embedding methods, that it can reduce the noise when input into the probabilistic graphical model.

The reason why we still need CRF instead of just using dense layer right after the BiLSTM layer. Lets consider the scenario in named entity recognition, all the I-[tag] should followed by B-[tag]. Using dense layer we cannot promise this rule. But with CRF, because there is no chance (zero probability) for a named entity with I-[tag] before a B-[tag], that is the transition probability on the transition matrix will be zero, thus it guarantees this rule establish.

In practice, I use dropouts to make the model more generalize that I use dropout wrapper on the RNN cell and set the dropout rate with 0.5 while training. And do not forget to disable the dropouts while doing inference.

## 3 Experiment

As the requirement of the tasks. I split the labeled data into two part. 70% for training set and 30% for test set. The experiment result are the test on the test set. (And finally will use the entire labeled data to train another model and use it to label final submission).

### 3.1 Word Segmentation Evaluation

The evaluation result check out Table 1. We can found that the BiLSTM with CRF model is perform much better than pure CRF model.

[6] https://en.wikipedia.org/wiki/Conditional_random_field

[7] https://en.wikipedia.org/wiki/Hidden_Markov_model

| Model | Parameters | | | | Test Set | | |
|---|---|---|---|---|---|---|---|
| | Input Embedding | Train Epoch | Learing Rate | Dropout Rate | P | R | F1 |
| CRF | One-hot | 100 | 0.01 | None | 76.36 | 75.06 | 75.70 |
| BiLSTM + CRF | One-hot | 10 | 0.01 | 0.5 | 90.02 | 91.08 | 90.55 |

Table 1: Best evaluation result of word segmentation on the test set (in %)

### 3.2 Named Entity Recognition Evaluation

The evaluation result check out Table 2. The training parameter is basically the same except the learning rate. This part will be discussed in the next section.

| Model | Parameters | | | | Test Set | | |
|---|---|---|---|---|---|---|---|
| | Input Embedding | Train Epoch | Learing Rate | Dropout Rate | P | R | F1 |
| CRF | One-hot | 100 | 0.001 | None | 61.37 | 59.32 | 60.32 |
| BiLSTM + CRF | One-hot | 10 | 0.001 | 0.5 | 85.59 | 80.47 | 82.95 |

Table 2: Best evaluation result of named entity recognition on the test set (in %)

## 4 Analysis and Discussion

The performance of pure CRF is not good enough. As my speculation, that is because one-hot is not good enough as a word representation method for this kind of task. That is why introducing BiLSTM acting the word representation character will gain a big improvement.

I have ask my previous teammate, he said he tried TF-IDF and also the fastText Chinese pre-trained embedding but just worse than one-hot encoding. But he replaced the BiLSTM layer with BERT and get the even better result. It shows that maybe we a dynamic embedding model is better than just look up a embedding vector table.

The pure CRF has become much more tricky in the named entity recognition task. That because of the imbalance of the label (almost 90% is N tag), thus even the word-level accuracy is very high, but found out the model learned nothing and can only output N tags or a few other tags but wrong.

Thus I deprecate CRF as named entity recognition model and use BiLSTM with CRF model instead. But found out it just can't perform as well as word segmentation. And finally found that the key point is the learning rate. When I lower the learning rate for named entity recognition (about $\frac{1}{10}$ times) it become much better.

I was planned to try different embedding approach as input and the other parameter settings but the computation time is much longer than I expected thus I was only able to test with some configuration after debuging the first valid model. I was extremely relied on the Intel DevCloud[8] (20 nodes with 6 cores 3.4GHz CPU and 180GB memory) in my previous project. But it was running out of the six months trail few weeks ago. And training one epoch of BiLSTM model on my laptop will need 10 hours, it make the experiment much harder.

Hope I can keep improve the model after this course when found a substitute or just power enough calculation resource. I think these two tasks are quite basic but also very meaningful.

[8] https://devcloud.intel.com/datacenter/

And also maybe I can warp it as an application by calling the API that I have left for the extension. The source code is on my GitHub[9] .

Finally, thanks teacher for the teaching of this semester. It helps me a lot and even spending very much times on this course and doing projects but it is totally worth it.

# References

[1] Huang, Z., Xu, W., and Yu, K. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[2] Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[9] https://github.com/daviddwlee84/DeepLearningPractice/blob/master/Project/CWSNER